Appendix A 10/774,913

Draft revised ITU-T Recommendation H.264

Advanced video coding for generic audiovisual services

Editor's Note:

This document contains (in integrated form)

- Errata report corrections relative to the May 2003 standard (ITU-T Rec. H.264 | ISO/IEC 14496-10) up to and including the disposition of comments to the March 2004 meeting in Munich (starting with JVT-K051r1).
- Fidelity range extensions amendment as of the July 2004 meeting in Redmond in Draft Amendment 1, which was not separately submitted for approval in ITU-T (the non-integrated version is JVT-L047d12 and the integrated version is JVT-L050d5).
- Additional error report corrections reflecting the outcome of the October 2004 meeting in Palma de Mallorca, Spain (with the non-integrated version being JVT-M049d6).
- Additional error report edits reflecting the outcome of the January 2005 meeting in Hong Kong (subclause 7.4.2.11 semantics of rbsp_stop_one_bit and rbsp_alignment_zero_bit changing "is a single bit equal to" to "shall be equal to", and subclauses 73.5.1 and 7.4.5.1 syntax and semantics of intra_chroma_pred_mode changing u(v) to ue(v) and specifying its range of values).

Author(s) or Contact(s):

Gary Sullivan
Microsoft Corporation
One Microsoft Way
Redmond, WA 98052 USA
Tel: +1 (425) 703-5308
Fax: +1 (425) 706-7329
Email: garysull@microsoft.com

Thomas Wiegand
Fraunhofer Institute for Telecommunications, Heinrich-Hertz-Institut
Einsteinufer 37, D-10587 Berlin, Germany
Tel: +49 (30) 31002-617
Fax: +49 (30) 392 72 00
Email: wiegand@hhi.de

Ajay Luthra
Motorola Corporation
6420 Sequence Drive
San Diego, CA 92121 USA
Tel: +1 (858) 404 3470
Fax: +1 (858) 404 2501
Email: aluthra@motorola.com

[REFERENCE: JVT-N050d1wcm.doc]

Appendix A 10/774, 913

Title page to be provided by ITU-T | ISO/IEC

DRAFT INTERNATIONAL STANDARD
Draft Revised ISO/IEC 14496-10 (E)
Draft Revised Rec. H.264 (E)
DRAFT ITU-T RECOMMENDATION

TABLE OF CONTENTS

Foreword xiii

0 Introduction xiv
0.1 Prologue xiv
0.2 Purpose xiv
0.3 Applications xiv
0.4 Publication and versions of this specification xiv
0.5 Profiles and levels xiv
0.6 Overview of the design characteristics xv
0.6.1 Predictive coding xv
0.6.2 Coding of progressive and interlaced video xv
0.6.3 Picture partitioning into macroblocks and smaller partitions xvi
0.6.4 Spatial redundancy reduction xvi
0.7 How to read this specification xvi

1 Scope 1

2 Normative references 1

3 Definitions 1

4 Abbreviations 8

5 Conventions 9
5.1 Arithmetic operators 9
5.2 Logical operators 10
5.3 Relational operators 10
5.4 Bit-wise operators 10
5.5 Assignment operators 10
5.6 Range notation 10
5.7 Mathematical functions 10
5.8 Variables, syntax elements, and tables 11
5.9 Text description of logical operations 12
5.10 Processes 13

6 Source, coded, decoded and output data formats, scanning processes, and neighbouring relationships 13
6.1 Bitstream formats 13
6.2 Source, decoded, and output picture formats 13
6.3 Spatial subdivision of pictures and slices 19
6.4 Inverse scanning processes and derivation processes for neighbours 19
6.4.1 Inverse macroblock scanning process 19
6.4.2 Inverse macroblock partition and sub-macroblock partition scanning process 20
6.4.2.1 Inverse macroblock partition scanning process 21
6.4.2.2 Inverse sub-macroblock partition scanning process 21
6.4.3 Inverse 4x4 luma block scanning process 21
6.4.4 Inverse 8x8 luma block scanning process 22
6.4.5 Derivation process of the availability for macroblock addresses 22
6.4.6 Derivation process for neighbouring macroblock addresses and their availability 22
6.4.7 Derivation process for neighbouring macroblock addresses and their availability in MBAFF frames 23
6.4.8 Derivation processes for neighbouring macroblocks, blocks, and partitions 24
6.4.8.1 Derivation process for neighbouring macroblocks 25
6.4.8.2 Derivation process for neighbouring 8x8 luma block 25
6.4.8.3 Derivation process for neighbouring 4x4 luma blocks 26
6.4.8.4 Derivation process for neighbouring 4x4 chroma blocks 26
6.4.8.5 Derivation process for neighbouring partitions 26

- Otherwise (ctxIdxOffset is equal to 77), the following applies.
  - The derivation process for neighbouring macroblocks specified in subclause 6.4.8.1 is invoked and the output is assigned to mbAddrA and mbAddrB.
  - Let the variable condTermFlagN (with N being either A or B) be derived as follows.
    - If mbAddrN is available and mb_type for the macroblock mbAddrN is equal to I_PCM, condTermFlagN is set equal to 1
    - Otherwise, if any of the following conditions is true, condTermFlagN is set equal to 0
      - mbAddrN is not available or the macroblock mbAddrN is skipped
      - binIdx is equal to 0 and CodedBlockPatternChroma for the macroblock mbAddrN is equal to0
      - binIdx is equal to 1 and CodedBlockPatternChroma for the macroblock mbAddrN is not equal to2
    - Otherwise, condTermFlagN is set equal to 1.
  - The variable ctxIdxInc is derived as

$$ctxIdxInc = condTermFlagA + 2 * condTermFlagB + ((binIdx == 1) ? 4 : 0) \quad (9\text{-}5)$$

NOTE – When a macroblock uses an Intra_16x16 prediction mode, the values of CodedBlockPatternLuma and CodedBlockPatternChroma for the macroblock are derived from mb_type as specified in Table 7-11.

9.3.3.1.1.5 Derivation process of ctxIdxInc for the syntax element mb_qp_delta

Output of this process is ctxIdxInc.

Let prevMbAddr be the macroblock address of the macroblock that precedes the current macroblock in decoding order. When the current macroblock is the first macroblock of a slice, prevMbAddr is marked as not available.

Let the variable ctxIdxInc be derived as follows.

- If any ofthe following conditions is true, ctxIdxInc is set equal to0
  - prevMbAddr is not available or the macroblock prevMbAddr is skipped
  - mb_type of the macroblock prevMbAddr is equal to I_PCM
  - The macroblock prevMbAddr is not coded in Intra_16x16 prediction mode and both CodedBlockPatternLuma and CodedBlockPatternChroma for the macroblock prevMbAddr are equal to0
  - mb_qp_delta for the macroblock prevMbAddr is equal to 0
- Otherwise, ctxIdxInc is set equal to 1.

9.3.3.1.1.6 Derivation process of ctxIdxInc for the syntax elements ref_idx_l0 and ref_idx_l1

Input to this process is mbPartIdx.

Output of this process is ctxIdxInc.

The interpretation of ref_idx_lX and Pred_LX within this subclause is specified as follows.

– If this process is invoked for the derivation of ref_idx_l0, ref_idx_lX is interpreted as ref_idx_l0 and Pred_LX is interpreted as Pred_L0.

– Otherwise (this process is invoked for the derivation of ref_idx_l1), ref_idx_lX is interpreted as ref_idx_l1 and Pred_LX is interpreted as Pred_L1.

Let currSubMbType be set equal tosub_mb_type[mbPartIdx].

The derivation process for neighbouring partitions specified in subclause 6.4.8.5 is invoked with mbPartIdx, currSubMbType, and subMbPartIdx = 0 as input and the output is assigned to mbAddrA\mbPartIdxA and mbAddrB\mbPartIdxB.

With ref_idx_lX[mbPartIdxN] (with N being either A or B) specifying the syntax element for the macroblock mbAddrN, let the variable refIdxZeroFlagN be derived as follows.

- If MbaffFrameFlag is equal to 1, the current macroblock is a frame macroblock, and the macroblock mbAddrN is a field macroblock

refIdxZeroFlagN = ((ref_idx_lX[mbPartIdxN] > 1) ? 0 : 1) (9-6)

- Otherwise,

refIdxZeroFlagN = ((ref_idx_lX[mbPartIdxN] > 0) ? 0 : 1) (9-7)

Let the variable predModeEqualFlag be specified as follows.

- If the macroblock mbAddrN has mb_type equal to P_8x8 or B_8x8, the following applies.
  - If SubMbPredMode(sub_mb_type[mbPartIdxN]) is not equal to Pred_LX and not equal to BiPred, predModeEqualFlag is set equal to 0, where sub_mb_type specifies the syntax element for the macroblock mbAddrN.
  - Otherwise, predModeEqualFlag is set equal to 1.
- Otherwise, the following applies.
  - If MbPartPredMode(mb_type, mbPartIdxN) is not equal to Pred_LX and not equal to BiPred, predModeEqualFlag is set equal to 0, where mb_type specifies the syntax element for the macroblock mbAddrN.
  - Otherwise, predModeEqualFlag is set equal to 1.

Let the variable condTermFlagN (with N being either A or B) be derived as follows.

- If any of the following conditions is true, condTermFlagN is set equal to0
  - mbAddrN is not available
  - the macroblock mbAddrN has mb_type equal to P_Skip or B_Skip
  - The macroblock mbAddrN is coded in Intra prediction mode
  - predModeEqualFlag is equal to 0
  - refIdxZeroFlagN is equal to 1
- Otherwise, condTermFlagN is set equal to 1.

The variable ctxIdxInc is derived as

ctxIdxInc = condTermFlagA + 2 * condTermFlagB (9-8)

9.3.3.1.1.7 Derivation process of ctxIdxInc for the syntax elements mvd_l0 and mvd_l1

Inputs to this process are mbPartIdx, subMbPartIdx, and ctxIdxOffset.

Output of this process is ctxIdxInc.

The interpretation of mvd_lX and Pred_LX within this subclause is specified as follows.

- If this process is invoked for the derivation of mvd_l0, mvd_lX is interpreted as mvd_l0 and Pred_LX is interpreted as Pred_L0.
- Otherwise (this process is invoked for the derivation of mvd_l1), mvd_lX is interpreted as mvd_l1 and Pred_LX is interpreted as Pred_L1.

Let currSubMbType be set equal to sub_mb_type[mbPartIdx].

The derivation process for neighbouring partitions specified in subclause 6.4.8.5 is invoked with mbPartIdx, currSubMbType, and subMbPartIdx as input and the output is assigned to mbAddrA\mbPartIdxA\subMbPartIdxA and mbAddrB\mbPartIdxB\subMbPartIdxB.

Let the variable compIdx be derived as follows.

- If ctxIdxOffset is equal to40, compIdx is set equal to 0.
- Otherwise (ctxIdxOffset is equal to 47), compIdx is set equal to 1.

Let the variable predModeEqualFlag be specified as follows.

- If the macroblock mbAddrN has mb_type equal to P_8x8 or B_8x8, the following applies.

- If SubMbPredMode(sub_mb_type[mbPartIdxN]) is not equal to Pred_LX and not equal to BiPred, predModeEqualFlag is set equal to 0, where sub_mb_type specifies the syntax element for the macroblock mbAddrN.
- Otherwise, predModeEqualFlag is set equal to 1.

- Otherwise, the following applies.
  - If MbPartPredMode(mb_type, mbPartIdxN) is not equal to Pred_LX and not equal to BiPred, predModeEqualFlag is set equal to 0, where mb_type specifies the syntax element for the macroblock mbAddrN.
  - Otherwise, predModeEqualFlag is set equal to 1.

Let the variable absMvdCompN (with N being either A or B) be derived as follows.

- If any of the following conditions is true, absMvdCompN is set equal to 0
  - mbAddrN is not available
  - the macroblock mbAddrN has mb_type equal to P_Skip or B_Skip
  - The macroblock mbAddrN is coded in Intra prediction mode
  - predModeEqualFlag is equal to 0
- Otherwise, the following applies
  - If compIdx is equal to 1, MbaffFrameFlag is equal to 1, the current macroblock is a frame macroblock, and the macroblock mbAddrN is a field macroblock

    absMvdCompN = Abs(mvd_lX[mbPartIdxN][subMbPartIdxN][compIdx]) * 2 (9-9)

  - Otherwise, if compIdx is equal to 1, MbaffFrameFlag is equal to1, the current macroblock is a field macroblock, and the macroblock mbAddrN is a frame macroblock

    absMvdCompN = Abs(mvd_lX[mbPartIdxN][subMbPartIdxN][compIdx]) / 2 (9-10)

  - Otherwise,

    absMvdCompN = Abs(mvd_lX[mbPartIdxN][subMbPartIdxN][compIdx]) (9-11)

The variable ctxIdxInc is derived as follows

- If (absMvdCompA+ absMvdCompB) is less than 3, ctxIdxInc is set equal to 0.
- Otherwise, if (absMvdCompA + absMvdCompB) is greater than 32, ctxIdxInc is set equal to 2.
- Otherwise ((absMvdCompA + absMvdCompB) is in the range of 3 to32, inclusive), ctxIdxInc is set equal to 1.

9.3.3.1.1.8 Derivation process of ctxIdxInc for the syntax element intra_chroma_pred_mode

Output of this process is ctxIdxInc.

The derivation process for neighbouring macroblocks specified in subclause 6.4.8.1 is invoked and the output is assigned to mbAddrA and mbAddrB.

Let the variable condTermFlagN (with N being replaced by either A or B) be derived as follows.

- If any of the following conditions is true, condTermFlagN is set equal to0
  - mbAddrN is not available
  - The macroblock mbAddrN is coded in Inter prediction mode
  - mb_type for the macroblock mbAddrN is equal to I_PCM
  - intra_chroma_pred_mode for the macroblock mbAddrN is equal to0
- Otherwise, condTermFlagN is set equal to 1.

The variable ctxIdxInc is derived by

ctxIdxInc = condTermFlagA + condTermFlagB (9-12)